Confidence in ASCI Scientific Simulations

James A. Ang and Timothy G. Trucano, Sandia National Laboratories, Albuquerque, NM
David R. Luginbuhl, U. S. Department of Energy, Washington, DC

Abstract

The U. S. Department of Energy's (DOE) Accelerated Strategic Computing Initiative (ASCI) program calls for the development of high end computing and advanced application simulations as one component of a program to eliminate reliance upon nuclear testing in the U.S. nuclear weapons program. This paper presents results from the ASCI program's examination of needs for focused validation and verification (V&V). These V&V activities will ensure that 100 TeraOP-scale ASCI simulation code development projects apply the appropriate means to achieve high confidence in the use of simulations for stockpile assessment and certification.

We begin with an examination of the roles for model development and validation in the traditional scientific method. The traditional view is that the scientific method has two foundations, experimental and theoretical. While the traditional scientific method does not acknowledge the role for computing and simulation, this examination establishes a foundation for the extension of the traditional processes to include verification and scientific software development that results in the notional framework known as Sargent's Framework. This framework elucidates the relationships between the processes of scientific model development, computational model verification, and simulation validation.

This paper presents a discussion of the methodologies and practices that the ASCI program will use to establish confidence in large-scale scientific simulations. While the effort for a focused program in V&V is just getting started, the ASCI program has been underway for a couple of years. We discuss some V&V activities and preliminary results from the ALEGRA simulation code that is under development for ASCI. The breadth of physical phenomena and the advanced computational algorithms that are employed by ALEGRA make it a subject for V&V that should typify what is required for many ASCI simulations.

Background

The DOE Defense Programs Stockpile Stewardship Program (SSP) calls for judgment-based confidence as a necessary requirement for eliminating the need for nuclear testing in the U.S. nuclear weapons program [Larzelere98]. A simulation is considered *predictive* if it represents a major means of providing information to a weapons expert in all SSP applications relevant to the simulation and the primary (or only) means in specific critical stockpile situations. The goal for this ASCI V&V program is to develop simulations that are predictive in this sense.

The traditional scientific method is based on a process of observation, hypothesis development, experimental design, hypothesis test, and iterative improvement. A schematic framework for the traditional scientific method is shown in Figure 1. For hundreds of years this framework for science had two foundations, experimental and theoretical. Since the development of digital computers in the 1940's for the Manhattan Project, we now recognize a third foundation, computational science.

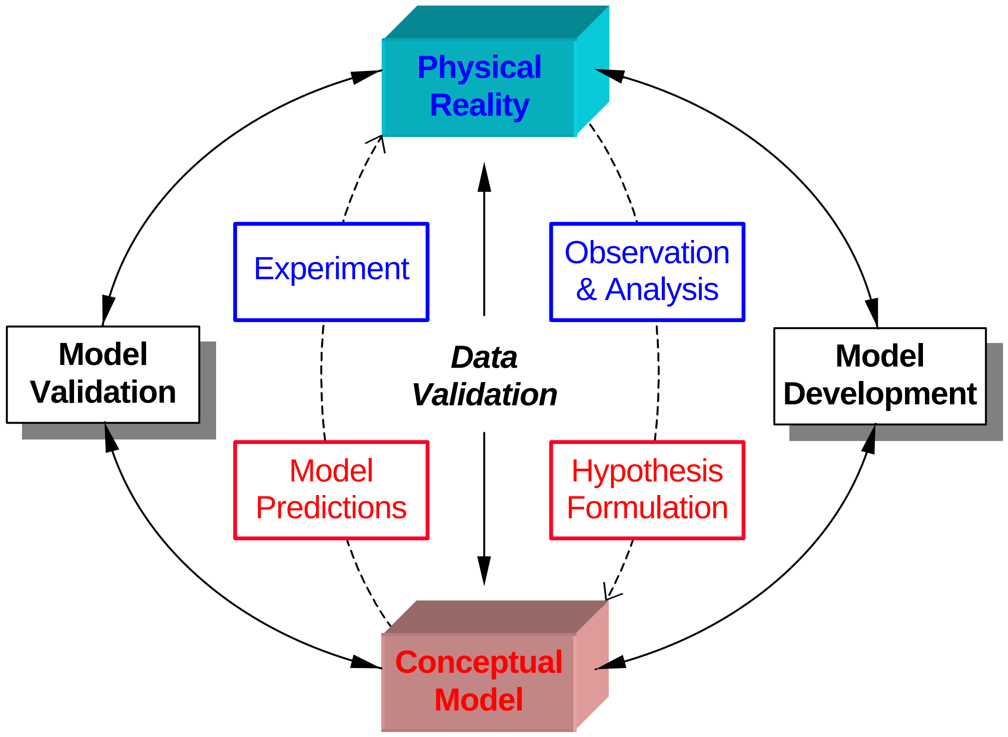

Figure 1. A framework for the traditional scientific method. This diagram illustrates the iterative nature of the scientific method and the traditional role for the validation process.

The processes of validation and verification deal respectively with the interfaces between computational simulations and experimental observations, and computer models and theoretical models. To illustrate the interactions between these three foundations of science it is useful to examine the scientific framework shown in Figure 2. This diagram is based on Sargent’s Framework [Knepell93]. While Sargent's Framework was developed by the operations research community, this diagram provides useful guidance in the processes we can use to develop confidence in scientific simulations. In this framework there are three main objects: physical reality, conceptual models, and computer simulations, which are respectively, the subject or product of the experimental, theoretical, and computational sciences. The interface processes between these objects are shown in the outer ring, and they are supported by the activities shown in the inner boxes along the dashed lines. All of these objects, processes, and activities are governed by a common need for data validation, i.e., assessment of data quality; consistent units, common datum, etc. A comparison of these figures reveals that the process of model validation in the traditional scientific method framework in Figure 1 expands to encompass the code verification and simulation validation processes and the computer simulation object that are shown in the framework for the new scientific method in Figure 2. This is the impact of computers on the traditional scientific method.

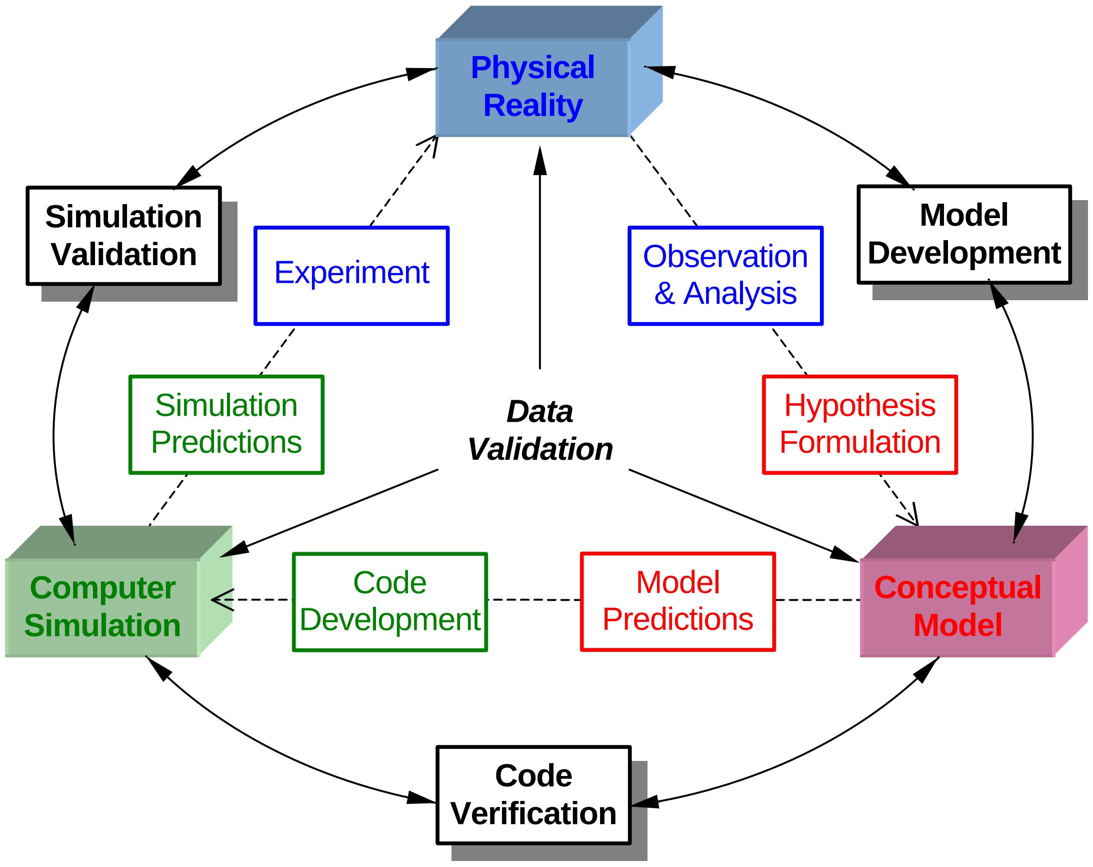

Figure 2. A framework for the new scientific method. This diagram illustrates how computational science interacts with the traditional activities of experimental and theoretical science.

For this paper we use the following definitions which are consistent with those of IEEE, AIAA, and the U. S. Defense Department's Defense Modeling and Simulation Office [IEEE82], [AIAA98], [DMSO96]:

Validation - the process of determining the degree to which a computer simulation is an accurate representation of the real world.

Verification - the process of determining that a computer simulation correctly represents the conceptual model and its solution.

"Verification and Validation" or *"Validation and Verification"*? It is common to see the former phrase because chronologically simulations are *verified* before they are *validated*. However, because in the ASCI V&V program the most visible process is validation, and a key element of this program is to coordinate with other DOE Defense Program efforts to obtain needed validation data, we have chosen to use the latter phrase.

Verification of Scientific Simulations

[Boehm81] has boiled the definition of software verification down to answering the question, "Are we building the software right?" In that regard, there are several components of a successful software verification program for ASCI codes. An integral part of establishing confidence in any computer simulation is to ensure that the software has been developed according to some set of standards and that the process for development ensures a proper transformation from conceptual model to finished code. The ASCI V&V program intends to establish a strong verification component along these lines.

This verification component will draw from accepted software engineering practices where possible, but we will also need to consider the unique aspects of both the domain of interest and the advanced computer architectures on which the simulations will be run. For the purposes of this paper, we will consider the "domain of interest" to be "scientific simulation," by which we mean simulations of the complex, physical processes of the real world. It could be argued that this is not specific enough to take into account all the simulations being developed in ASCI, but it will suffice for our discussion of verification below.

Software Engineering Practices

Obviously, there are many verification processes that transcend specific domains. Good software engineering processes apply whether one is developing a small database application or a large, complex, real-time embedded system. Indeed, scientific computing is hardly immune from the types of problems that result from the lack of a well thought-out V&V process. [Hatton97] documents some dramatic error statistics derived from his broad evaluation of scientific software.

Two principles taken from [DMSO96] are worth calling out to guide the V&V effort. The first principle is that "verification and validation ...should be an integral part of the entire ... life cycle." This principle is exemplified by Figure 3, adapted from [Ould96], which they use to "provide a framework for testing." Note in the figure that the output from each component activity of the development of the software corresponds directly to a deliverable during the testing of the software. One could infer from this view that planning for acceptance testing should take place during requirements analysis, and so on.

The key point is that verification efforts cannot take place only at the end of code development. Physical scientists and engineers have a reputation, maybe unearned, for not employing the most systematic of processes in code development (see [Wilson96], for example.) But "code and test, code and test" is not an effective approach to the development of highly reliable code, especially when the stakes are as high as they are in ASCI.

The second guiding principle, which derives from the first, is that "verification and validation ... must be planned and documented." An integral part of ASCI's V&V effort is the requirement for each development team to develop a detailed V&V plan.

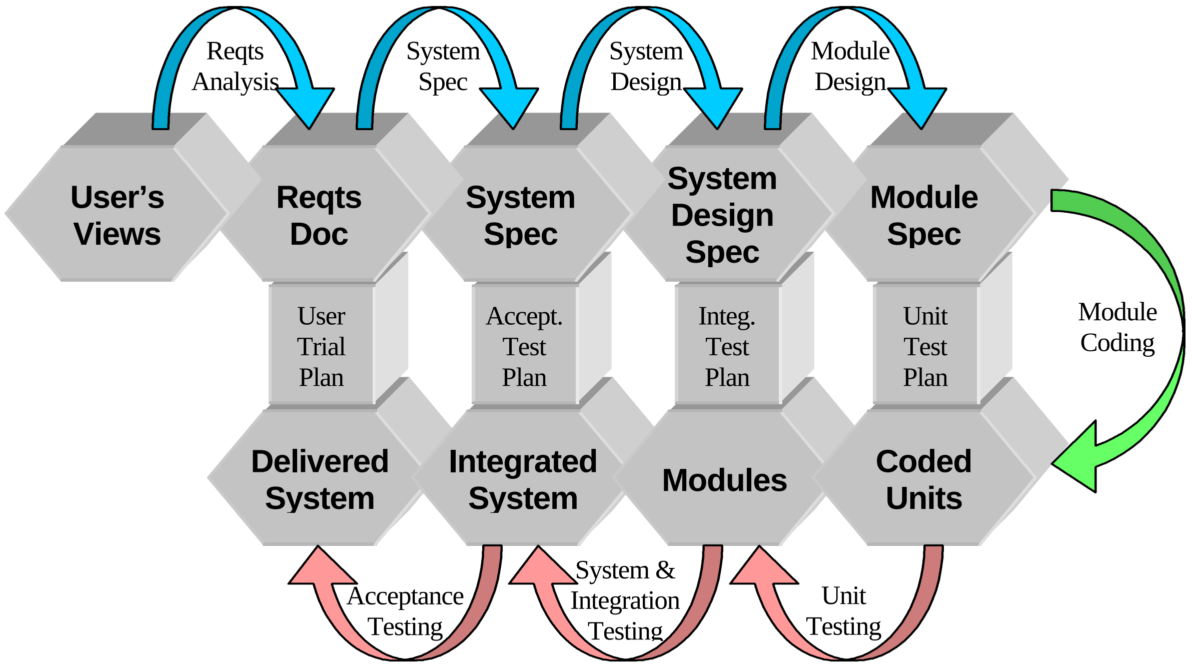

Figure 3. A lifecycle approach to software development and testing [Ould86].

Verification Techniques

Most of the "tried and true" techniques for software verification should certainly be considered for verifying ASCI software. Space limitations prevent a discussion of the entire litany of techniques available, but some techniques merit special mention. We will use the high-level taxonomy of V&V techniques adopted by [DMSO96]: informal, static, dynamic, and formal techniques.

Informal Techniques

Many informal techniques, such as walkthroughs, reviews, and inspections, should be easily adaptable to ASCI software development, having been proven over the past twenty-five years to add significant value to all classes of large software projects [O'Neill97]. These "people-oriented" approaches to evaluation are applicable at any stage of software development. The challenge in evaluating ASCI software at this level is to ensure the proper level of domain expertise (i.e., the physics of nuclear weapons) among the reviewers.

Static Techniques

Static techniques (e.g., fault analysis, semantic and syntactic analysis, structural analysis) are probably the easiest to introduce into the ASCI software development process. Static analysis lends itself to automation, so to the extent that ASCI has adopted these tools, static analysis is already accepted as a part of ASCI V&V (indeed, compilers themselves comprise one avenue of static analysis, i.e., syntactic evaluation (DMSO96]). The value of static techniques is made all the more apparent in [Hatton97]. His static analysis of a variety of scientific software revealed dramatic results in terms of faults per thousand lines of code. One point of interest: "nuclear engineering" codes were among the worst in number of faults.

Dynamic Techniques

Dynamic techniques involve actually exercising the software developed and include various types of testing (integration testing, acceptance testing, regression testing), debugging, and assertion checking. The key to all good testing (and this applies no less to ASCI codes) is in test planning as described in the second guiding principle above. Thus, for example, acceptance test development should begin as requirements for the software are created. Further, integration tests should arise out of the definition of module interfaces.

Regression tests are of special interest in ASCI since software is expected to run on several different platforms, each with a potentially different architecture. Regression tests are necessary to ensure the software is still correct when ported to a new platform. Regression testing also aims to control a common problem in developing scientific software. *Improvements* in one physics module might lead to an error occurring in a different module.

In fact, computer architecture is a special consideration for the tools that implement static and dynamic techniques (debuggers, for instance). The subtleties of running a code on a large number of coordinated computer nodes must be accounted for in the development of static analysis tools. Even for dynamic analysis tools, repeatability of results could become problematic, if problem setup does not assign the same computations to the same nodes each time.

Formal Techniques

Formal verification methods are the most difficult to introduce in any development effort. Combinatorial explosion usually presents a formidable barrier to a comprehensive approach to formal methods, thus the size and complexity of ASCI codes becomes a factor. To this we add the difficulty of applying techniques based on discrete mathematical structures to a simulation of continuous physical processes.

One approach to addressing the problem of size is to consider whether there might be critical portions of the software to which some sort of formal verification technique might be applied. This still leaves us with the problem of modeling continuous mathematics with discrete structures. An avenue of research and future application might be to consider the application of hybrid automata [Henzinger96], which attempt to combine continuous mathematics into finite structures, to scientific software.

Validation of Scientific Simulations

Historically, design and engineering codes were used as tools to support new weapons development and nuclear testing within the U. S. nuclear weapons program. These previous computational simulation and modeling capabilities for nuclear weapons were empirically based and, due to computer hardware limitations, most weapon assessments were restricted to two-dimensional or crudely zoned three-dimensional models.

With the cessation of U. S. nuclear testing and no requirements for new weapon production, the role of computational simulation and modeling has changed. As the stockpile ages, changes in weapon components and materials will require improved physical models and simulations. Efforts to extend the lifetime of the stockpile rely heavily on our ability to perform predictive, integrated, three-dimensional computer simulations to assess the stockpile. Predictive three-dimensional simulation codes require improved physics, materials, and engineering models that must be validated by experienced weapon scientists and engineers.

Within the new simulation-based SSP environment, a primary driver for the experimental program is to provide validation data that will be used to establish confidence in simulation capabilities which, in turn, will be used to certify weapons. This confidence will be established by exercising the feedback loop between simulations and experiments as shown in the upper left-hand portion of Figure 2. Experiments must not only assist in the certification of components and systems, they now must be designed to provide the necessary information to establish confidence in predictive simulation codes over a broad parameter space. Conversely, new experiments will also serve to establish regions in physical parameter space where confidence in simulation results is degraded. This new requirement calls for increased fidelity in the collection of experimental data to exercise the regimes of physics that our more powerful and predictive three-dimensional codes can provide. Thus, our framework is shifting from certification of point designs to predictive confidence in a larger design space.

Requirements for Validation Data

The comparison and analysis of an ASCI simulation with a nuclear event or weapon stockpile-to-target-sequence test requires access to a large amount of data. This includes not just the results of the test diagnostics, but detailed information about all of the device parts, details of the diagnostic system, and data on non-nuclear calibration experiments. As indicated in the middle of Figure 2, all of these data need a careful assessment of their quality.

Physics, materials, and engineering models are needed for the SSP. Data to validate these models will come from a variety of sources. Physics and theoretical models that reduce reliance on empirical approximations will need experiments specifically designed to validate these models, as well as calibrate key parameters used in their implementation. Measurements made in the process of monitoring the health of the U. S. nuclear stockpile will be required to validate materials aging models needed to predict the lifetime of weapons components. Manufacturing process simulations will require data from manufacturing operations and require interfaces to design simulations to support concurrent engineering design. Past nuclear and non-nuclear test data that include test failures and mysteries, tests that have large sensitivity to physics and materials models, and tests that had different weapon component manufacturing processes and techniques will provide valuable validation data to establish the highest confidence in future weapons simulations.

The V&V teams for these codes will be composed of code developers, software engineers, analysts, theoreticians, experimentalists and designers, as well as experienced users of weapons simulation codes, to insure that effective and relevant V&V is carried out. Meeting stockpile assessment and certification requirements will naturally lead to code user acceptance and confidence.

The V&V program derives its requirements from existing stockpile drivers to help us identify the specific validation tasks required to establish confidence in our computational capabilities for stockpile assessment and certification. We expect that some data that will be required will be of a different form and fidelity than we have historically required. Experimental validation data needs will include:

- High-fidelity measurements of material physical and chemical properties that are used as inputs to simulation codes
- High-fidelity experiments for validation of code sub-models/models and coupled phenomena through laboratory and facility tests
- Integrated code validation, where possible, through comparison against past nuclear test data, full system tests and stockpile surveillance data

Types of Validation Data

There are several sources of data for validation of ASCI simulations. They will be discussed below as archived data from nuclear tests, archived data from non-nuclear tests, fundamental physics experimental data, integrated system certification tests, and stockpile surveillance data. Obviously data from past experiments and certification tests could be considered part of the archive of non-nuclear tests. For this paper, we consider these separate categories to focus on new validation data that could be obtained by providing guidance and recommendations to the relevant experimental and archiving programs for future data collection opportunities.

Archived Data from Nuclear Tests

The archive of data from nuclear tests is a unique resource for validation of codes to simulate nuclear weapons performance and safety, and components and system reliability in hostile nuclear environments. This body of data provides an important link to the historical test-based certification process, and thus is a key to assuring confidence in the transition to a simulation-based certification process. Work on V&V should generate useful input to nuclear weapons archiving projects on what data are most important for validation.

An important element of validation is robustness to develop confidence in predictions for a range of problems, not just for the validation test suite. This is particularly needed for validation against nuclear test data because systems of interest are unlikely to match a specific test configuration. Confidence in robustness can be gained by using the same code to perform calculations and compare results with data from a number of nuclear tests.

Archived Data from Non-Nuclear Tests

Assessments of nuclear performance of devices depend not only on evaluation against archival nuclear data, but also on types of archival non-nuclear data. During development of a weapon system, a series of weaponization tests characterized the full-system response to storage, transportation, and stockpile-to-target-sequence requirements. And finally, the physics underlying the code implementations relies on a large database of physical data characterizing the material behaviour in the weapon.

As with data from nuclear tests, the engineering simulations can be partially validated against small-scale and component tests. However, demonstrating confidence in engineering system simulations to address stockpile stewardship issues usually requires validation against large-scale system tests. Examples are integrated system flight tests, hostile x-ray environment tests, and large-scale target penetration tests. Types of measurements from these tests for validation of simulations of the nuclear explosive package include acceleration, stress, strain, and damage histories and measurements of delivered system function. Non-nuclear component test examples include arming, fuzing, and firing response to hostile x-ray or abnormal fire environments, and re-entry vehicle response to deployed vibration and thermal conditions.

Another large body of archival data is the physical properties database underlying all of the nuclear and non-nuclear modeling. The physical material property data (EOS, opacity, cross sections, constitutive data) also requires assessment based upon the requirements for validation stated above. The codes used to evalute and generate these physical databases will also require validation just as the hydro, nuclear and engineering/weaponization codes.

Fundamental Physics Experimental Data

New experiments will be required to test ASCI simulations in three-dimensional regimes for which previous data are not appropriately detailed, or does not exist. Current experimental capabilities have evolved to include more extensive or more accurate diagnostics on traditional types of experiments as well as the development of new kinds of experiments generating completely new types of data. For example, traditional diagnostics on certain shockwave physics tests have been extended to cover a more complete range of the performance regime of devices and new diagnostics have been added to measure specific physical phenomena with greater accuracy. Completely new kinds of experiments have extended the utility of shockwave physics experiments in characterizing complex behaviour. An example is the possible application of advanced holographic techniques to collect three-dimensional experimental data at very high resolutions [Ang93].

Coupled computational/experimental activities will play an important role in the design and optimization of these high-fidelity experimental programs to ensure that experimental conditions and instrumentation are appropriate for the generation of the data that is required for code validation. Examples of laboratory-scale experiments include those that capture spatially, temporally, and spectrally resolved data to compare with multi-dimensional simulation results. Simulation results can also be used to design lab-scale experiments to capture data around boundary layers, critical points, and other loci of relevant physics. Laboratory scale experiments at new kinds of facilities such as gas gun, high energy laser, and pulsed-power experiments provide new types of data in physical regimes relevant to nuclear weapons, and provide opportunities for understanding physical phenomena in three-dimensional and dimensionally fine detail not available during weapon development.

System Certification Test Data

The V&V program will provide the ability to identify and follow through on opportunities for adding diagnostics and instrumentation to full system certification tests that would help capture unique data that might otherwise be lost for validation purposes. With the increased complexity of the required experiments, due to the need for higher fidelity data, and restrictions on the number of laboratory and full system tests, the role of simulation in the design of experiments is more important. Simulation results can be applied to estimate data acquisition requirements by providing experimenters with diagnostics requirements for instrumentation dynamic range, sensor locations/orientations, data rates, etc.

Stockpile Surveillance Data

Data collection is of particular importance in assessing the condition of aged materials, components, and systems in the stockpile. Modeling of aging in materials has been limited at least partly by a lack of aging data. The ASCI materials aging application development teams have ongoing interactions with the stockpile surveillance program that is monitoring the health and status of the U. S. nuclear weapons stockpile. The V&V program will build on these interactions to obtain the data that we need to validate our aging simulations. These aging simulations can point out problems to look for, and may even suggest measurement techniques and ranges for future surveillance activities.

Needs for Comparison of Simulation Results with Validation Data

High consequence computing is one of the implications of simulation-based stockpile stewardship. Because of this key role, the V&V program needs to move aggressively from the past paradigm of code calibration, i.e., adjustment of modeling parameters to attain agreement with experimental data, to a more rigorous paradigm of *validation science*.

Validation science, which relies upon the development and deployment of a host of advanced tools and methodologies, will aim to accomplish the following objectives:

- Encourage the application of techniques for improved computationally based, quantitative experimental data - calculated data comparisons. The same goal can be achieved for code-to-code comparisons using these tools.
- Achieve improved and quantitative understanding of the selection of data most appropriate for the code validation task.
- Determine more precise and quantitative implications of accurate or inaccurate computational comparison with the selected experimental data for the code validation task.
- Minimize the impact of human error on the computational component of the SSP through application of the code validation process.

Uncertainty Quantification

One approach that can provide a systematic foundation for validation science is *uncertainty quantification* (UQ). This is the technology for quantitatively estimating the uncertainty in the output of code calculations from characterizations of input uncertainty, including underlying model uncertainties and uncertainty in the existing validation database. One of the results of applying UQ is that confidence in the results of code calculations can be quantitatively assessed.

In addition, UQ can serve as an organizing principle for the code-experimental data comparison activity. For example, UQ provides intrinsic data about parts of a calculation that contain the greatest degree of uncertainty. It is then logical to drive the validation activity, for that particular application, with those experimental data that are most relevant to that uncertainty contribution. If these data do not exist, this also provides a rigorous rationale for guiding new experimental activity. UQ also suggests rigorous strategies for extrapolating code confidence from specific, successful data comparisons. We anticipate that additional work will have to be performed to make UQ as effective as possible in the ASCI code validation activity.

We will discuss UQ in greater detail below when we discuss specifics of the V&V activities associated with the ALEGRA code.

Challenge Problems

Ultimately, the development of *confidence* in the predictive capability of ASCI simulations will rely on more than just the systematic, rigorous validation and verification of codes and underlying physical models. Real confidence will only be achieved through the application of ASCI simulations to make predictions of “Challenge Problems” [McMillan96]. Confidence evolves by making iterations around the *new scientific method* framework in Figure 2. The V&V program will also work with the experimental elements of the SSP to establish structured challenge problems that provide an opportunity for published (within the stockpile stewardship community) pretest predictions.

A carefully designed challenge problem component will be significant to completing the shift from test-based to simulation-based certification. The use of simulations to make pretest predictions that can subsequently be validated or refuted with new experimental test results *involves risk*. In fact, the expectation is that challenge problems will lead to instances of spectacular simulation failures, but these failures are analogous to test failures at the Nevada Test Site (NTS). The potential for success or failure at the NTS provided an important cultural and sociological aspect to the initiation of nuclear designers with nuclear testing that validated designers as well as their designs [Gusterson96]. The transition from test-based stockpile stewardship to simulation-based stockpile stewardship must acknowledge this issue. Challenge problems will form the new proving grounds for the next generation of designers and weapons analysts.

Often, we learn more from our failures than our successes. [Petroski85] even notes this fact in the subtitle of his popular book, *To Engineer is Human*. The eventual outcome of simulation failures will be improvements in the simulations and ultimately, increased confidence in both the validated domain of a simulation capability and increased understanding of its limits. Note that if challenge problems never lead to simulation failures, we need to go back and find more difficult problems. It is by overcoming challenge problems that we will gain demonstrable confidence in our simulations.

V&V of the ALEGRA application

Introduction

ALEGRA V&V activities use a mixture of formal and informal procedures, rigorous constraints, and a wide variety of verification and validation problems. We will attempt to capture the scope of our thought about these components below, rather than attempt a complete discussion of the specifics. The reader should be aware that we are still attempting to learn how to transform some of these admittedly abstract concepts into concrete methodologies applicable to ALEGRA. In this sense there is a very real research component in the ALEGRA V&V project.

ALEGRA Summary

A brief summary of ALEGRA can be found in [Summers96]. As discussed there, ALEGRA is a multi-material, arbitrary-Lagrangian-Eulerian (ALE) strong shock wave physics code under development at Sandia National Laboratories. It combines the features of modern Eulerian shock codes with modern finite element Lagrangian codes. Basic finite element ALE shock hydrodynamics is also supplemented by a mesh refinement project based on finite element h-adaptivity principles. In addition to the basic shock wave physics algorithms, a variety of coupled physics capabilities are included in the code development effort, including coupled electro-mechanical response, magneto-hydrodynamics, and radiation transport. Successful application of the code requires more than accurate implementation of solution algorithms for these kinds of physics. A variety of accurate material models - equations of state, constitutive and fracture models, thermal conduction behavior, radiative opacity, electrical resistivity, plasma transport coefficient, piezoelectric and ferroelectric material descriptions, and others - must also be implemented, tasks which can easily be as daunting as the fundamental algorithm development when extreme levels of accuracy are required.

ALEGRA is written predominantly in the C++ programming language. We have also incorporated various Fortran-based models and libraries. The basic development of ALEGRA is performed on a variety of serial workstations. The code development team targets distributed-memory massively parallel (MP) computers for its primary applications platforms, including ASCI class MP computers. Approximately fifteen people currently write code for this project. The primary development environment is the SPARCworks environment on SUN workstations.

To better understand the approach of the ALEGRA project to validation, it is important to emphasize that ALEGRA is in some regards a tightly woven umbrella for a variety of physics code development projects. This is shown in Figure 4. The key projects that should be recognized for this discussion are the strong shock wave physics ALE core project, a mesh refinement project called HAMMER, a coupled electro-mechanics project called EMMA, and a radiation-magneto-hydrodynamics project called HYRUM. The object-oriented paradigm that runs throughout the project is essential for managing such a complex project effectively. The recognition of the physics dependencies and independence of these projects is also critical for understanding the bottom-up and top-down paths for validation in this environment. We will return to this in more detail in the validation section below.

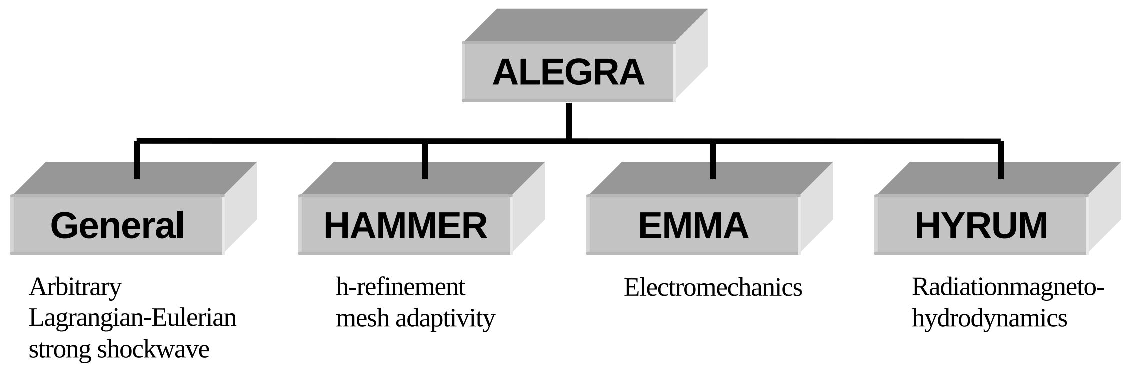
ALEGRA
General
HAMMER
EMMA
HYRUM
Arbitrary Lagrangian-Eulerian strong shockwave
h-refinement mesh adaptivity
Electromechanics
Radiationmagneto-hydrodynamics

Figure 4. ALEGRA is a set of projects under rather tightly integrated development.

Verification Approach for ALEGRA

In terms of the verification techniques described above, the ALEGRA project is pursuing the following approaches:

- The application of static and dynamic fault analysis tools for assessing program correctness. For example, we use a commercially available tool set for performing static and dynamic fault checking, as well as support tools for testing and test management, provided by Rational Software

Corporation [Rational98]. In particular, we use one tool available in this suite for static fault testing called Purify® [Rational98]. This is a comprehensive run-time code error and memory leak detection product.

- We are interested in tools to better inform our decisions regarding test problem suites. We are interested in extending our understanding of coverage to include "path" coverage. We discuss this issue in greater depth below.
- We are interested in safe language subsets and a recommended "style" for floating point C++ implementations of the complexity of ALEGRA (over one hundred-fifty thousand lines of C++, perhaps as much as one million lines of auxiliary code written in Fortran and C). To our knowledge there have been no discussions of safe C++ subsets appropriate for floating point implementations along the lines of *Safer C* [Hatton95].
- We are currently applying the ClearDDTS™ configuration and change management and bug tracking system for managing code bug reports and their resolution. This is a commercial web-based tool [Rational98].
- We seek useable "standards" for software engineering that are applicable to ASCI scale scientific software. At this time, the ALEGRA project is not working to formal standards, such as ISO 9000 [Sanders94] or the Software Engineering Institute Capability Maturity Model (CMM℠) [Ginsberg95].
- We use informal verification techniques. However, some of the more rigorous methods, such as formal software inspections, are used only irregularly. It is quite common on large scientific code projects like ALEGRA to rarely use formal software inspections. This is not a statement that we believe that such inspections are not useful, but rather a statement of historical fact. Systematic design and code walkthroughs and reviews have been regularly applied when major architectural changes or additions to ALEGRA have been suggested.
- No formal verification techniques are applied in the ALEGRA project, except those that may be implicit in tools like Purify™.
- Finally, we are also concerned with hardware reliability issues. At this time, there is no formal hardware testing performed by the ALEGRA project as a rigorous part of the testing process. This will likely change at some point in the future, but it is not clear at this time exactly what will be done. Under this we also include issues about the verifiability of operating systems and system libraries, as well as language environments.

Verification Test Problems and Procedures

Dynamic techniques for verification remain the most important component of the ALEGRA verification process. Ideally, we would like to define a series of test problems or procedures, each of which tests a well-characterized block of code, here simplistically referred to as a "module." The schematic representation of this is seen in Figure 5, where we show a selected test problem logically connected to a set of modules. We would like the modules arranged in order of importance, although how to do this is not especially clear in many cases. There are two ways that these dependencies can be made more elaborate, resulting in more systematic testing of the code implementation. In the first case, we can refine the granularity of the modules. For example, we might recognize that Module 1 contains 4 other key sub-modules. We could then attempt to design problems that test those sub-modules, which moves us towards a “component testing” framework. We refer to this as increasing the precision of the test problems that accomplish this. The effect of this is to refine Figure 5 into a far more complex tree of code modules. The ultimate conclusion of such an approach would be to have a test problem for every executable line of code, an obviously infeasible and undesirable strategy. The art (and we would like to make it a science) is to strike the best balance between detail and effort. As always, the implied goal is not so much some kind of “proof” that the implementation is correct, as it is a continuing improvement of the correctness of the code.

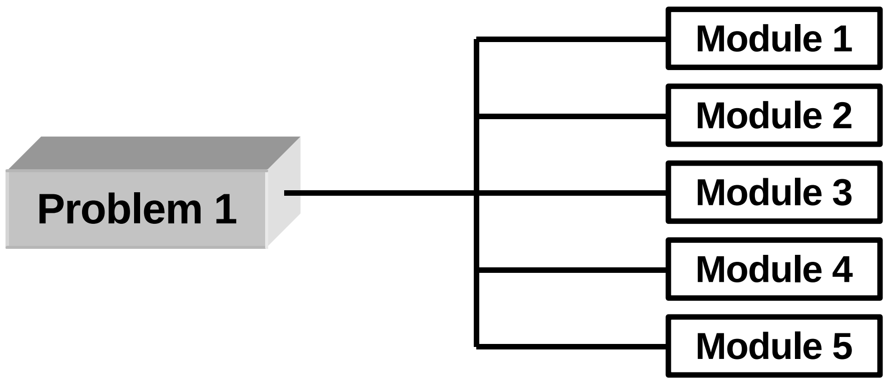

Figure 5. Schematic representation of the module dependence for a verification test problem.

As a second strategy, we might also consider the possibility of testing flow paths through the code. In other words, the diagram in Figure 5 would then expand to be a network diagram of perhaps amazing complexity, in which the flow paths among the modules are also suggested. We would then seek to rank order the flow paths by importance as well.

Combining both views - module granularity and flow path characterization - leads to rather improbable effort for testing the code. Nonetheless, this is a logically correct way to test ALEGRA. The key to being able to perform anything like this testing is to have appropriate metrics for measuring module and path importance, and to have tools that allow us to determine coverage. A rather unrefined zeroth order approach for assessing importance is to simply accept that the code architects and developers have a good notion of the most important modules to test first. Flow paths is a rather more difficult topic, which we will not address in this paper. To assess coverage, ALEGRA utilizes PureCoverage™ [Rational98], which at least provides information about software coverage provided by a given test or set of tests. It is not clear that this tool in itself allows us to easily quantitatively assess either particularly fine levels of granularity or flow paths.

Regression Testing

ALEGRA utilizes regression testing with a suite of around 100 problems. Recall [IEEE82], that regression testing is used to test implementation stability. The regression test suite is applied before code check-ins, to insure that the regression tests provide numbers that are the same as prior to the implementation of the new code. This is straightforward in principle, but somewhat difficult to manage in the context of ALEGRA, especially for an MP environment. In addition, applying PureCoverage™ we have found that we are currently only covering approximately 45% of the code with our regression test suite. It is obvious at this point that to achieve greater logical coverage, as well as to control the potential for exponential growth of CPU time spent in regression testing, we must apply a component testing framework, hence moving to finer granularity in this testing.

It is wise to keep in mind that "stability" in the meaning of regression testing is a neutral concept: regression testing asks less about the actual accuracy of the answers to the regression tests than it does about their invariance under new code implementation. This conflicts somewhat with the overall goals of a physics code development project to constantly improve solutions. The ideal world would be to have only regression tests that are also correct in a frozen sense. In practice this is somewhat surprisingly difficult to achieve for a code such as ALEGRA. A much broader suite of test problems is required.

Verification Problems for Scientific Codes

Verification problems properly demand total concern with accuracy of solution and implementation. Standard general classes of verification problems for ALEGRA include (1) "analytic" test problems, (2) problems which test known constraints (geometric symmetry, physical conservation laws, boundary conditions, and so on) and (3) careful and controlled comparisons with other computer simulations for more complex problems. We utilize all three general classes of verification problems.

We have a wide variety of verification problems, varying from problems that are close to component or unit tests to rather integral test problems. These problems tend to be aligned along the lines of the ALEGRA subproject diagram in Figure 4. Verification problems exist for all of the major categories of the project, the total number being on the order of 150 at this time. Enforcement of diagrams like Figure 5 for each of these problems is rather weak at this time. We intend to strengthen our understanding of the module links with time.

One component of the suite of verification problems is simply a closure of the regression test suite to the point where we know that all of the problems are giving correct answers. There are simple distinctions that may be helpful. A regression test problem may run a problem for only a few cycles, while its full verification counterpart may run the problem to a fixed time at which comparison with an analytic or other code solution can be performed. Also, a regression test may use a less demanding version of a problem, while the verification version may be the most demanding version of the problem that we can define. We typically are far less concerned with the amount of time it takes to execute verification problems. Verification problems are typically executed in regular fashion, but they are not necessarily executed upon code check-in every time.

In an attempt to do a better job of defining "importance" (as mentioned above), we are attempting to rank verification problems in importance. We are beginning to use an arbitrary scale of 1 (least important) to 4 (most important) to weight problems. We are also attempting to quantify our understood performance on given problems using the same scale: 1 (poor) to 4 (good). It is not easy to assign either importance or performance on certain problems. We are advocating the use of groups (perhaps containing a code developer, an "expert" who could be an ALEGRA analyst or simply someone experienced in the use of codes for solving similar problems, and a peer reviewer) to help establish these measures for given problems. Much of our concern revolves around our observations that complex code projects similar to ALEGRA can often waste effort on solving selected verification problems with increasing fidelity that turn out to be relatively unimportant as implementation tests. While tools - such as coverage analysis - may help us, we feel that it is important to at least attempt to establish consensus about what the most important problems might be and what current ALEGRA performance on those problems is. This then contributes to our ability to more rapidly focus improvement efforts on only important problems. This is also a theme that we will emphasize in our discussion of validation problems below.

Sensitivity Analysis

We view sensitivity analysis - quantitative measures of the sensitivity of the key variables in a solution to code inputs and algorithmic parameters - as a tool that can contribute to assessing importance. For example, there are many potential ways of testing the ability of ALEGRA to calculate a strong shock wave in a plasma. In a "real application", a sensitivity analysis might reveal that the solution depends most importantly on only a few selected parameters. Combined with a sufficiently detailed coverage analysis, this information can provide a map of the most relevant modules, flow paths, and the most important parameter dependencies. Verification problems that have similar use of these modules, that have similar flow paths, and that have similar sensitivities to parameters can then be weighted to be more important for testing implementations for this application than verification problems that are poorer approximations to these constraints. In addition, quantitatively understanding sensitivities allows us to logically refine the implication of given verification problems, thus increasing their apparent leverage on implementation correctness issues.

Stating this is one thing, but implementing tools that allow us to realize this vision is something else again. There are several approaches to performing sensitivity analysis, both deterministic and non-deterministic. It is beyond the scope of this article to discuss this in detail. However, we state that we are currently implementing a tool [Tong97] that will begin to provide us with sensitivity analysis information using a non-deterministic framework. We feel that this also happens to be an advantage for going the next step and addressing uncertainty quantification (see Validation below).

Validation Approach for ALEGRA

Our overall approach to validation in ALEGRA is governed by the general requirements stated above:

- A logical separation must be maintained between activities and goals aimed at verification and those aimed at validation.
- Experimental data are required for validation of ALEGRA.
- Code comparisons are problematical at best when used for validation of ALEGRA.
- Explicit intent to "certify" or "accredit" the use of ALEGRA for its intended applications is a goal of the validation process. We remind that reader that a useful definition of certification is "Certification is the formal decision that a model or simulation is acceptable for use for a specific purpose." [DMSO96] The goal of certification is completely dependent upon the intended application and is, therefore, very narrow in scope.
- Validation should be aimed at breaking ALEGRA, as well as determining that ALEGRA is accurate within specified regimes. It is easier to accomplish this task when a high performance experimental program is integrated within the structure of the code development project. We currently have experimental activities at Sandia associated with ALEGRA validation that provide this kind of scope throughout the entire project diagram in Figure 4.

- We are engaged in the process of aligning our validation requirements with the data definitions given above. This requires refining the detailed breakdown of stockpile requirements for ALEGRA into validation requirements, an effort that is progressing at the current time.

Our approach to performing validation on ALEGRA is quite similar in spirit to our approach for verification. The focus of our validation concern is now on physics, however, rather than implementations. A diagram similar to that in Figure 5 can be drawn for validation, in which problems induce connections to "physics" rather than modules. We have included such a diagram for emphasis in Figure 6. Again, this diagram implies that the physics connections are somehow rank ordered in terms of importance. The issue of granularity also emerges here. Typically, coarse granularity suggests a focus on a more integral kind of physics. Finer granularity suggests passing to more specific physics.

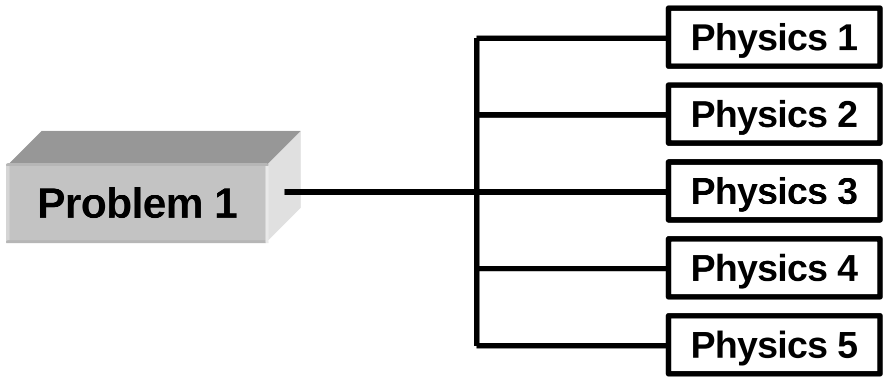

Figure 6. Schematic representation of the physics dependence for a validation test problem.

Validation Problems

The largest part of our validation process is studying validation problems, so we will focus the remainder of our discussion on this aspect. A validation problem study is actually a complex and typically difficult study, which requires a careful assessment of a set of experimental data as well as reasoned conclusions about the physical quality of a calculation. We believe that a team of people is required to do this right. A reasonable group of people would include: a problem owner, one analyst (or more) responsible for performing ALEGRA simulations of the problem, a code developer, a person capable of writing an acceptance requirement that bears on requirements for the quality of the simulations, and a peer review person (preferably from another code project who is expert in simulations directly or indirectly related to the problem under discussion) for benchmarking. It would also be worthwhile to require the experimenter who generated the data to participate in this team. We assume that such a group of people could spend from 6 months to one year fully assessing the performance of ALEGRA on a given validation problem.

We are in the process of defining a variety of validation problems, based on the above-mentioned validation requirements, for all of the major physics components of the code. A very high level view of the organization of these problems is most conveniently conveyed in terms of where we can acquire data. Validation data for the strong shock wave physics component includes data from time-resolved shock wave physics experiments performed at Sandia and other laboratories, classic compressible gas dynamics experimental data, data from a variety of military ordnance applications, including long-rod penetrators and conventional explosives, and hypervelocity impact data recommended by the hypervelocity impact community. These data basically provide an ensemble of shock wave data that encompass most of the dynamic compression and release regime that ALEGRA is concerned with and which is accessible to laboratory experiments. We are also in the process of generating specific data of interest for particular applications through specially designed experiments at Sandia. Certain data are available for the electro-mechanics component of ALEGRA, and will also continue to be collected in the future. As far as the radiation-magneto-hydrodynamics component, a prime provider of validation data here is the Sandia fast Z-pinch programs. Challenge problem activities will evolve from current experimental efforts at Sandia.

Integral versus Specific Validation Data

There is an important balance that must be attained between integral and specific validation tests. For example, in ALEGRA an integral application might be to model the explosion of a supernova. Integral applications are often the most important applications for scientific codes. This places great weight on finding validation problems, and reasonable data, that operate in such a regime. However, integral problems are also weak in and of themselves because they do not effectively validate in a top-down manner. In other words, it would be very difficult to infer the reality of the shock wave calculations in ALEGRA from comparisons with light curve data from real supernovae. Thus, the necessity for more focused shock wave data becomes clear. Unfortunately, such data will likely be available only in pressure regimes that are orders of magnitude smaller than the relevant pressures in a supernova. Exactly how to proceed in the face of this discrepancy is perplexing. ALEGRA has no magic bullet solution - we simply stress that a reasonable validation process must at least direct rigorous attention to these issues. This illustrates why we believe that a requirement for any validation activity is that there be experimental attempts to refute the predictions of the code, as well as to confirm them.

The same issues of weight and performance that we raised in regard to verification problems for ALEGRA are relevant to validation problems. They are more complex to understand, however. In particular, there are often significant problems in coming to a consensus as to how accurately a given set of experimental data may have been modeled. Even if the comparison of ALEGRA with experimental data could be reduced to some kind of numerical metric, it would be misleading to think of such a metric strictly as an issue of numerical accuracy. This is because there are many uncertainties that enter complicated calculations which seek to model complex experiments. We will touch on uncertainty quantification below as an important technology for validation.

Uncertainty Quantification

Uncertainty quantification for ALEGRA validation parallels the use of sensitivity analysis for verification. Simply put, uncertainty quantification has two problems that it studies. First, given model (code) uncertainties, what are the quantified uncertainties in the important outputs of simulations? This is sometimes referred to as the forward prediction problem. The uncertainties referred to are almost always treated using stochastic schemes of inference (in other words, statistical inference). Notice that one interesting and immediate consequence of this is that quality or accuracy assessments of the comparison of code results with data become stochastic. The implications of this for code validation in general, and ALEGRA validation in particular, are beyond the scope of this paper. However, realization of this fact is one of the prime reasons that we are beginning our implementation of sensitivity analysis capability in ALEGRA with a statistical approach. The same tools can be used to study forward prediction problems and we intend to use these tools in future validation studies for ALEGRA.

Second, one might wish to evaluate, and perhaps minimize, code uncertainties on the basis of knowledge of output uncertainties (as might be revealed in comparisons with experimental data). This is sometimes called the backward prediction problem, and is considered to be a harder problem than forward prediction. It is also of great significance when we concern ourselves with extracting the maximum specific validation content for physics treatments in a code from a class of integral data. It is safe to say that detailed thinking about the backward prediction problem in this context is essentially non-existent. Dealing with such problems is most likely years away from inclusion in formal ALEGRA validation activities.

An example of a recent validation study - related to ordnance applications of ALEGRA - can be found in [Carroll97]. This study took about six months to perform. Considerable effort was involved in attempting to assess the quality and uncertainty of the experimental data sets that were used, and the quantitative meaning of the code comparisons with the data. This task was not facilitated by the fact that the experimenter who performed the experiments was unavailable. In addition, we have not yet addressed the issue of how to actually peer review this work in a formal sense. Finally, none of the uncertainty quantification issues discussed above was treated in this paper. Our confidence in the ability of ALEGRA to simulate ordnance velocity long-rod penetrator events into monolithic metal definitely improved as a result of the study, however. This study has been followed by similarly complex studies of the ability of ALEGRA to successfully model data from explosively formed projectile experiments, as well as shaped-charge jet tests.

Planned ASCI V&V Activities and Final Observations

We are just getting started with the effort to establish validation and verification methodologies that can span all the ASCI application development efforts. Some of our codes like ALEGRA have developed V&V plans as part of their code development activities. In the coming year a new focused V&V Program will come on line to support V&V activities that go beyond the resources and responsibility of the individual ASCI application development teams.

Planned ASCI V&V Activities

Effective V&V Planning

Because V&V is so crucial to the development of high quality, predictive ASCI codes, it is important for each application development project to have a well-documented V&V process appropriate to the software development task. This process should address V&V activities that occur at all stages of software development, as illustrated in Figure 3. The accompanying V&V plan should at a minimum describe a clear set of V&V goals, explain the various techniques and tools to accomplish V&V (especially for verification), and describe the data from joint computational/experimental activities needed for code validation. A minimum goal of the V&V program should be to ensure that all code projects have appropriately developed V&V plans.

Evaluation of Current Standards

There is a need to re-evaluate common software engineering practices for application to scientific simulations that will be run on hierarchical, distributed memory, MP ASCI computer systems. An assessment of software development standards and their applicability to the ASCI is vital to ensuring that the codes are indeed being "built right." Examples of relevant software engineering practices include, but are not limited to those described in [AIAA98], [DMSO96], [IEEE82], and [Ginsberg95].

Process Improvement Tools and Techniques

The outcome of this activity should be a determination of tools, techniques, benchmarking practices and methodologies that are generally applicable to the verification of ASCI-scale scientific simulations. Each code development team could then integrate these high level standards and methodologies for their specific application.

For validation, process improvement means defining protocols for code usage, specifying techniques for comparing simulation results to validation data along with their associated uncertainties and errors, and specifying validation data requirements to measure the predictive nature of the codes. A key role for the V&V Program will be to collect, organize, and communicate our needs for validation data to the organizations and programs that own the validation data.

Independent Technical Reviews

While it is expected that the ASCI code projects will incorporate V&V into their own development processes, an additional measure of independence is necessary. One means for achieving this independence is through a course of program-wide software technical reviews. These would involve a series of walkthroughs, reviews, and inspections, as outlined above, and it might involve, for example, cross-lab peer review or even external review.

Challenge Problems

Challenge problems will be needed to exercise the predictive capabilities of the various ASCI scientific simulations. The definition of challenge problems and the development of a process for running challenge problems will require input from code developers, simulation users and experimentalists from the three DOE defense program national laboratories. The capabilities of new DOE SSP experimental facilities will be factored into the ability to collect test data in new physical regimes. These facilities, combined with opportunities to carry out tests on existing facilities with new materials in new configurations offers many opportunities for staging pre-test predictions followed by collection of validation data.

Final Observations

We conclude with some observations on the relationship of validation to confidence in simulations. First, to claim some rigorous content from a validation problem, we must avoid the use of research models in our code. By definition, research models are not validated. This in turn implies rather tight control on the kinds of models that will ever be allowed in a *validated* code. If the code is accredited for a specific application, this also likely means that no new models of any kind can be allowed in the code without a complete repeat of the accreditation process. This is difficult to enforce in a code that is designed to implement advanced technologies, whether model, algorithm, or programming based.

Second, we ask whether a *validated* code in some sense guarantees reliable simulations. The obvious answer is no, because the analyst who uses the code may make an error in pre-processing, code execution, or post-processing. This implies that if the goal of validation is to improve our confidence in the reliability of a code calculation, then formal steps must be taken to deal with the fact that humans perform these calculations. The answer to this problem is not for a code project to claim that they are not responsible for incorrect use of the code. We generically refer to this issue as “calculation protocols.” Two effective techniques to include within calculation protocols are, the use of formal input inspections (identical to software inspections, but applied to user code inputs) and formal training. A third highly likely technique is to use multiple users for really important calculations, followed by elicitation methodologies for establishing a consensus analysis.

Acknowledgments

The authors developed much of this paper in conjunction with planning for a new focused ASCI effort in V&V. This planning activity afforded the authors many opportunities for exchange of ideas with colleagues from Lawrence Livermore, Los Alamos and Sandia National Laboratories. Specific individuals who have contributed to the development of ideas contained in this paper include: Tom Adams, Don McCoy, Juan Meza, Bob Thomas, Bill Oberkampf, Paul Hommert, Cynthia Nitta, Richard Klein and Gary Carlson. Responsibility for the formulation and presentation of ideas remains with the authors and do not constitute official policy statements of either the U. S. Department of Energy or Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

References

[AIAA98] American Institute of Aeronautics and Astronautics, Standards Department, *Guide for the Verification and Validation of Computational Fluid Dynamics Simulations*, AIAA G-077-1998, 1998.

[Ang93] Ang, James A., Bruce D. Hansche, Carl H. Konrad, William C. Sweatt, Scott G. Gosling, and Randy J. Hickman, “Pulsed Holography for Hypervelocity Impact Diagnostics”, *International Journal of Impact Engineering*, Vol. 14, 1993, pp. 13-24.

[Boehm81] Boehm, Barry W., *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, NJ, 1981, p.37.

[Carroll97] Carroll, Daniel E., Eugene S. Hertel, Jr., and Timothy G. Trucano, “Silsby Long-Rod Calculations,” Sandia National Laboratories, SAND97-2765., 1997.

[DMSO96] Office of the Director of Defense Research and Engineering, *Verification, Validation, and Accreditation (VV&A) Recommended Practices Guide*, Defense Modeling and Simulation Office, Washington, DC., November 1996.

[Ginsberg95] Ginsberg, Mark P., “Process Tailoring and the Software Capability Maturity Model,” Carnegie Melon University Software Engineering Institute Technical Report CMU/SEI-94-TR-024, 1995.

[Gustafson98] Gustafson, John, "Computational Verifiability and Feasibility of the ASCI Program," *IEEE Computational Science and Engineering*, Vol. 5, No. 1: January-March 1998, pp. 36-45.

[Gusterson96] Gusterson, Hugh, *Nuclear Rites: A Weapons Laboratory at the End of the Cold War*, University of California Press, Berkeley, 1996.

[Hatton95] Hatton, Les, *Safer C: Developing Software for High-Integrity and Safety-Critical Systems*, McGraw-Hill, London, 1995.

[Hatton97] Hatton, Les, "The T Experiments: Errors In Scientific Software," *IEEE Computational Science and Engineering*, Vol. 4, No. 2: April-June 1997, pp. 27-38.

[Henzinger96] Henzinger, Thomas A. "The theory of hybrid automata," Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996), pp. 278-292.

[IEEE82] IEEE Standard for Software Verification and Validation Plans, IEEE Standards Board, 1982.

[Knepell93] Knepell, Peter L. and Deborah C. Arangno, *Simulation Validation: A Confidence Assessment Methodology*, IEEE Computer Society Press, Los Alamitos, 1993.

[Larzelere98] Larzelere, Alexander R., "Creating Simulation Capabilities," *IEEE Computational Science and Engineering*, Vol. 5, No. 1: January-March 1998, pp. 27-35.

[McMillan96] Private communication with Charles McMillan, Lawrence Livermore National Laboratory, 1996.

[O'Neill97] O'Neill, Don, "Software Inspections," *Software Technology Review*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. June 1997, pp. 353-359.

[Ould86] Ould, Martyn A. and Charles Unwin, eds., *Testing in Software Development*, The British Computer Society Monographs in Informatics, Cambridge University Press, Cambridge, 1986.

[Petroski85] Petroski, Henry: *To Engineer is Human: The Role of Failure in Successful Design*, St. Martin's Press, New York, 1985.

[Rational98] See http://www.rational.com, Rational Software Corp., Boulder, CO, 1998.

[Sanders94] Sanders, Joc and Eugene Curran , *Software Quality*, Addison-Wesley, Reading, 1994.

[Summers96] Summers, Randall M., James S. Peery, Michael W. Wong, Eugene S. Hertel, Jr., Timothy G. Trucano, and Lalit C. Chhabildas, "Recent Progress In ALEGRA Development and Application to Ballistic Impacts," in Proceedings of the 1996 Hypervelocity Impact Symposium, to be published in *International Journal of Impact Engineering*, 1996.

[Tong97] Tong, Charles H. and Juan C. Meza, "DOOMSDACE: A Distributed Object-Oriented Software System with Multiple Samplings for the Design and Analysis of Computer Experiments," Sandia National Laboratories draft report, 1997.

[Wilson96] Wilson, Gregory V., "What Should Computer Scientists Teach to Physical Scientists and Engineers?," *IEEE Computational Science and Engineering*, Vol. 3, No. 2: Summer 1996, pp. 46-55.